REMARKS/ARGUMENTS

Claims 1-3, 5-9, 11-15, 17, and 18 remain pending in this application. Claims 1, 3, 7, 9, 13, and 15 have been amended, and claims 4, 10, and 16 have been cancelled. No new matter has been added to the prosecution of this application. For at least the reasons stated below, Applicant asserts that all claims are now in condition for allowance.

CLAIM REJECTIONS UNDER 35 U.S.C. § 102

Claims 1-3, 5-9, 11-15, 17, and 18 are rejected under 35 U.S.C. § 102(e) as being anticipated by *Chang et al.*, U.S. patent No. 6,157,953. Applicant respectfully opposes these rejections. Applicant asserts that not every element of every claim is taught by the reference. MPEP § 2131 provides:

> "A claim is anticipated only if each and every element as set forth in the claim is found, either expressly or inherently described, in a single prior art reference." *Verdegaal Bros. v. Union Oil Co. of California*, 814 F.2d 628, 631, 2 USPQ2d 1051, 1053 (Fed. Cir. 1987). "The identical invention must be shown in as complete detail as is contained in the ... claim." *Richardson v. Suzuki Motor Co.*, 868 F.2d 1226, 1236, 9 USPQ2d 1913, 1920 (Fed. Cir. 1989). The elements must be arranged as required by the claim...

The present invention generally provides for a method for maintaining a security profile throughout nested service invocations on a distributed, component-based system, including the following elements:

(a) providing interconnections between distributed components each having nested service invocations;
(b) identifying a user;
(c) associating the user with roles;
(d) creating a user context instance upon successful identification of the user, wherein the user context instance includes information about the user including the roles and a unique user identifier;
(e) receiving a request from the user to invoke a first service on a first component, wherein the first component invokes a second service of a second component such that the user context instance is passed as a parameter from the first component to the second component, and wherein completion of the second service is necessary to complete the first service;
(f) querying the user context instance for the unique user identifier;
(g) comparing the unique user identifier in the user context instance with an access control list for verifying that the user has access to the first component; and
(h) comparing the unique user identifier in the user context instance with an access control list for verifying that the user has access to the second service of the second component.

Because not every element of every claim is taught by the reference, the Examiner's § 102 rejections are unsupported by the art and should be withdrawn.

User Context is Passed as a Parameter from the First Component to the Second Component

The present invention provides for creating a user context instance that includes, *inter alia*, a unique user identifier. Then, in response to the user request to invoke a first service on a first component, the user context instance is passed as a parameter from the first component to a second component. This aspect of the present invention facilitates the comparison of the unique user identifier contained in the user context instance with an access control list to verify that the user has access to the second service. Further, the claims have been amended to incorporate this aspect of the present invention into independent claims 1, 7, and 13.

In Applicant's invention, information about the user is represented in a shared user context object, which maintains a user's unique identification. *See* specification, p. 630, lines 1-3. The unique identification is checked against a resource's access control list (ACL). *Id.* When the appropriate conditions arise, such as successful user identification validation, an instance of the user context object is created, usually through a login routine. *See* specification, p. 630, lines 3-5. To facilitate the verification that the user has access to the first and second components being invoked, an instance of the user context is passed from component to component as a parameter of service invocations. *See* specification, p. 632, line 1-3; p. 630, lines 9-13. These aspects of the present invention are now reflected in claims 1-3, 5-9, 11-15, 17, and 18.

Chang, in contrast, describes a "method and apparatus of securing access to a service manager for the administration of services residing on multiple service host computers...," abstract, and is geared towards "automating the process of registering new applications and services at a central management location, such as a Web server, thereby reducing the amount of information the system administrator must remember and making a service available to end-users sooner," col. 5, ln. 39-44. However, *Chang* fails to teach creating an instance of a user context object that contains a unique user identifier and then passing that instance of the user context as a parameter from a first component to a second component.

Chang verifies a user's access by "comparing the user credentials or profile against the user's authentication and access control data in the database." Col. 13, lines 28-31. However, rather than storing the user information in a user context instance, as in the present claimed invention, *Chang* stores such user information in a database 212. *See* col. 12, lines 32-34. Further, *Chang* describes passing user credentials, after being verified, to service hosts, whereas the present invention passes a unique user identifier for the purpose of being verified. *See* col. 12, lines 34-40; col. 13, lines 41-46. *Chang* fails to teach creating an instance of a user context object that contains a unique user identifier and then passing that instance of the user context as a parameter from a first component to a second component.

First Component Invokes a Second Service on a Second Component

In the present claimed invention, each of the distributed components have nested service invocations. In particular, a first service is invoked on a first component and is invoked by the user

request, and a second service is invoked on a second component. Additionally, the claims require completion of the second service in order to complete the first service. Applicant draws the Examiner's attention to the specific relationships claimed: (a) the relationship between a component and its service (i.e. "first service" on the "first component," and the "second service" on the "second component"), and (b) the relationship between the first service/component and the second service/component (i.e. the first component invokes the second service).

Chang fails to describe these two aspects of the present invention: (1) completion of the second service is necessary to complete the first service, and (2) the first component, on which the first service is invoked and which is invoked by the user request, invokes the second service on the second component. The Examiner has asserted twice that these limitations are contained in *Chang*:

> In the Office Action dated August 8, 2002, Examiner asserts:
>
> > Chang teaches that when the user selects an instance of a service (first component), the user needs to enter information such as name and password to access the system, then the system compares the user credentials or profile against the user's authentication and access control data in the database (second component); the verification (second service) has to be performed before a connection (first service) is made (col. 7, lines 21-34; col. 13, lines 21-34).
>
> In the Advisory Action dated December 3, 2002, Examiner asserts:
>
> > When the user requests access to a service (first component) the system compares the user identification against the user's profile in the database (second component); then verification is performed and a connection is made.

Nowhere does *Chang* teach a single service/component (i.e. the "first service" invoked on the "first component") that both (1) depends on a second service for completion, and (2) is invoked by the user request and invokes the second service on the second component. Whether *Chang* provides for a first component and service, and a second component and service, is not presently in dispute. However, as shown below, *Chang* merely teaches a first service (connecting to service 218) on a first component (service 218) and a third component (console host 208) invoking a second service (verification) and a second component (database 212):

Claimed Invention	8/8/2002 Office Action	12/3/2002 Advisory Action
user request→ invokes first service on first component	user request→invokes first service (connecting to service, 218) on first component (service, 218)	user request→access to service (first component)
first component→ invokes second service on second component	**THIRD** component (console host, 208)→invokes second service (verification) on second component (DB, 212)	compare user ID against profile in database (second component)

In other words, *Chang* does not teach the first component invoking the second component, but rather some third component (i.e. the console host, 208), which was not requested by the user as claimed in the present invention, invoking the second component.

Specifically, *Chang* teaches selecting a service (218) (first component) from a service host (204/206) (col. 13, ln. 21-23; col. 6, ln. 1; Fig. 2). Next, *Chang* teaches that the verification (second service) performed on the database (212) (second component) is invoked by the console host (208) (**THIRD** component) via servlet CGI (228) and authentication layer (230) (col. 7, ln. 21-34; Fig. 2). This point is of the utmost importance because it is where *Chang* diverges from the present invention: the component in *Chang* that invokes the database, or "second component," is console host 208, and is not the same component invoked by the user request (the component invoked by the user request is service 218) as claimed in the present invention.

For at least the foregoing reasons, *Chang* does not set forth each and every element of the claims of the present invention. Further, *Chang* unequivocally does not described the "identical invention" "in as complete detail as is contained in the ... claim[s]" of the present invention as required by the Federal Circuit in *Richardson*. (emphasis added). Accordingly, Applicant respectfully requests that the Examiner's §102 rejections as to claims 1-3, 5-9, 11-15, 17, and 18 be withdrawn.

Additional Arguments as to Claims 5, 11, and 17

Dependent claims 5, 11, and 17 provide for the first service associating objects with the user context instance, wherein the object was created, updated, or deleted as a result of the invocation of the first service. *See* specification, p. 630, lines 9-16; Fig. 154. *Chang* describes using objects for storage, col. 15, lines 25-28, and receipt of data objects by a CPU, col. 16, lines 2-6. Contrary to Examiner's assertion that this limitation is taught by *Chang* at claim 1; col. 5, lines 22-44; col. 6, lines 52-59; col. 8, lines 37-48; col. 9, lines 43-55; and col. 10, lines 31-34, a review of these citations reveals that *Chang* makes no other mention of objects whatsoever, let alone association of objects with a user context. Nowhere does *Chang* describe that the "first service invoked associates any objects created, updated, or deleted as a result of the invocation of the first service with the user context instance" as set forth in claims 5, 11, and 17.

For these additional reasons, *Chang* further fails to show every element of dependent claims 5, 11, and 17.

CONCLUSION

Applicant submits that all pending claims are now allowable and respectfully requests that a Notice of Allowance be issued in this case. If the Examiner believes that a conference would be of value in expediting the prosecution of this application, the undersigned can be reached at the telephone number listed below.

Attached is a marked up version of the changes made to the specification by the current amendment. The attached page is captioned "Version with markings to show changes made." For Examiner's convenience, the pending claims have been attached hereto on the page captioned "Pending Claims."

Should any additional fees be necessary, the Commissioner is hereby authorized to charge or credit any such fees or overpayment to Deposit Account No. 50-1901 (Reference #60021-326501).

Respectfully submitted,

By _____

Christopher R. Hilberg, Reg. No. 48,740
Customer No. 29838

Oppenheimer Wolff & Donnelly LLP
1400 Page Mill Road
Palo Alto, CA 94304-1124
Telephone: 612.607.7386
Facsimile: 612.607.7100
E-mail: CHilberg@oppenheimer.com

Version with markings to show changes made
Serial No. 09/386,989

IN THE CLAIMS

Please cancel claims 4, 10, 16, and amend claims 1, 3, 7, 9, 13, and 15 as follows:

1. A method for maintaining a security profile throughout nested service invocations on a distributed, component-based system, comprising the steps of:

(a) providing interconnections between distributed components each having nested service invocations;

(b) identifying a user;

(c) associating the user with roles;

(d) creating a user context instance upon successful identification of the user, wherein the user context instance includes information about the user including the roles and a unique user identifier;

(e) receiving a request from the user to invoke a first service on a first component, wherein the first component invokes a second service of a second component such that the user context instance is passed as a parameter from the first component to the second component, and wherein completion of the second service is necessary to complete the first service;

(f) querying the user context instance for the unique user identifier ~~information about the user~~;

(g) comparing the unique user identifier in the user context instance ~~user information~~ with an access control list for verifying that the user has access to the first component; and

(h) comparing the unique user identifier in the user context instance ~~user information~~ with an access control list for verifying that the user has access to the second service of the second component.

3. A method as recited in claim 1, further comprising the step of modifying a user interface to provide access to actions that can be performed by the user based on the unique user identifier ~~an identity of the user~~ and the roles associated with the user.

7. A computer program embodied on a computer readable medium for maintaining a security profile throughout nested service invocations on a distributed, component-based system, comprising:

(a) a code segment that provides interconnections between distributed components each having nested service invocations;

(b) a code segment that identifies a user;

(c) a code segment that associates the user with roles;

(d) a code segment that creates a user context instance upon successful identification of the user, wherein the user context instance includes information about the user including the roles and a unique user identifier;

(e) a code segment that receives a request from the user to invoke a first service on a first component, wherein the first component invokes a second service of a second component such that the user context instance is passed as a parameter from the first component to the second component, and wherein completion of the second service is necessary to complete the first service;

(f) a code segment that queries the user context instance for the unique user identifier ~~information about the user~~;

(g) a code segment that compares the unique user identifier in the user context instance ~~user information~~ with an access control list for verifying that the user has access to the first component; and

(h) a code segment that compares the unique user identifier in the user context instance ~~user information~~ with an access control list for verifying that the user has access to the second service of the second component.

9. A computer program as recited in claim 7, further comprising a code segment that modifies a user interface to provide access to actions that can be performed by the user based on the unique user identifier ~~an identity of the user~~ and the roles associated with the user.

13. A system for maintaining a security profile throughout nested service invocations on a distributed, component-based system, comprising:

(a) logic that provides interconnections between distributed components each having nested service invocations;

(b) logic that identifies a user;

(c) logic that associates the user with roles;

(d) logic that creates a user context instance upon successful identification of the user, wherein the user context instance includes information about the user including the roles and a unique user identifier;

(e) logic that receives a request from the user to invoke a first service on a first component, wherein the first component invokes a second service of a second component such that the user context instance is passed as a parameter from the first component to the second component, and wherein completion of the second service is necessary to complete the first service;

(f) logic that queries the user context instance for the unique user identifier ~~information about the user~~;

(g) logic that compares the unique user identifier in the user context instance ~~user information~~ with an access control list for verifying that the user has access to the first component; and

(h) logic that compares the unique user identifier in the user context instance ~~user information~~ with an access control list for verifying that the user has access to the second service of the second component.

15. A system as recited in claim 13, further comprising logic that modifies a user interface to provide access to actions that can be performed by the user based on the unique user identifier ~~an identity of the user~~ and the roles associated with the user.

Pending Claims

Serial No. 09/386,989

1. A method for maintaining a security profile throughout nested service invocations on a distributed, component-based system, comprising the steps of:

(a) providing interconnections between distributed components each having nested service invocations;

(b) identifying a user;

(c) associating the user with roles;

(d) creating a user context instance upon successful identification of the user, wherein the user context instance includes information about the user including the roles and a unique user identifier;

(e) receiving a request from the user to invoke a first service on a first component, wherein the first component invokes a second service of a second component such that the user context instance is passed as a parameter from the first component to the second component, and wherein completion of the second service is necessary to complete the first service;

(f) querying the user context instance for the unique user identifier;

(g) comparing the unique user identifier in the user context instance with an access control list for verifying that the user has access to the first component; and

(h) comparing the unique user identifier in the user context instance with an access control list for verifying that the user has access to the second service of the second component.

2. A method as recited in claim 1, further comprising the step of logging all user interactions.

3. A method as recited in claim 1, further comprising the step of modifying a user interface to provide access to actions that can be performed by the user based on the unique user identifier and the roles associated with the user.

5. A method as recited in claim 4, wherein the first service invoked associates any objects created, updated, or deleted as a result of the invocation of the first service with the user context instance.

6. A method as recited in claim 1, wherein the user context instance encapsulates security certificates of the user.

7. A computer program embodied on a computer readable medium for maintaining a security profile throughout nested service invocations on a distributed, component-based system, comprising:

(a) a code segment that provides interconnections between distributed components each having nested service invocations;

(b) a code segment that identifies a user;

(c) a code segment that associates the user with roles;

(d) a code segment that creates a user context instance upon successful identification of the user, wherein the user context instance includes information about the user including the roles and a unique user identifier;

(e) a code segment that receives a request from the user to invoke a first service on a first component, wherein the first component invokes a second service of a second component such that the user context instance is passed as a parameter from the first component to the second component, and wherein completion of the second service is necessary to complete the first service;

(f) a code segment that queries the user context instance for the unique user identifier;

(g) a code segment that compares the unique user identifier in the user context instance with an access control list for verifying that the user has access to the first component; and

(h) a code segment that compares the unique user identifier in the user context instance with an access control list for verifying that the user has access to the second service of the second component.

8. A computer program as recited in claim 7, further comprising a code segment that logs all user interactions.

9. A computer program as recited in claim 7, further comprising a code segment that modifies a user interface to provide access to actions that can be performed by the user based on the unique user identifier and the roles associated with the user.

11. A computer program as recited in claim 10, wherein the first service invoked associates any objects created, updated, or deleted as a result of the invocation of the first service with the user context instance.

12. A computer program as recited in claim 7, wherein the user context instance encapsulates security certificates of the user.

13. A system for maintaining a security profile throughout nested service invocations on a distributed, component-based system, comprising:

(a) logic that provides interconnections between distributed components each having nested service invocations;

(b) logic that identifies a user;

(c) logic that associates the user with roles;

(d) logic that creates a user context instance upon successful identification of the user, wherein the user context instance includes information about the user including the roles and a unique user identifier;

(e) logic that receives a request from the user to invoke a first service on a first component, wherein the first component invokes a second service of a second component such that the user context instance is passed as a parameter from the first component to the second component, and wherein completion of the second service is necessary to complete the first service;

(f) logic that queries the user context instance for the unique user identifier;

(g) logic that compares the unique user identifier in the user context instance with an access control list for verifying that the user has access to the first component; and

(h) logic that compares the unique user identifier in the user context instance with an access control list for verifying that the user has access to the second service of the second component.

14. A system as recited in claim 13, further comprising logic that logs all user interactions.

15. A system as recited in claim 13, further comprising logic that modifies a user interface to provide access to actions that can be performed by the user based on the unique user identifier and the roles associated with the user.

17. A system as recited in claim 16, wherein the first service invoked associates any objects created, updated, or deleted as a result of the invocation of the first service with the user context instance.

18. A system as recited in claim 13, wherein the user context instance encapsulates security certificates of the user.